Application of an On-line Trainning Predictor/Controller to Dynamic Positioning of Floating Structures

Tzung-hang Lee[1], Yusong Cao[2] and Yen-mi Lin[1]

[1]*Department of Mechanical Engineering*
Tamkang University
Tamsui, Taipei, Taiwan 251, R. O. C.
E-mail: zouhan@mail.tku.edu.tw
[2]*School of Naval Architecture and Marine Engineering*
University of New Orleans
New Orleans, LA 70115, U. S. A.

Abstract

An on-line training functional-link neural network predictor/controller for dynamic positioning of water surface structures is described in this paper. To develop a neural network for time-evolving systems, the deterministic on-line training model in a traditional parameter identification theory and the functional-link network are combined. The system's previous input and output are used to be additional enhancements to the functional-link network. The on-line training neural network predictor acquires the knowledge about the system using a small number of samples of the latest system status measured on board of the structure. The trained functional-link neural network is used with an optimal controller to control the output of the system. The accuracy and robustness of the on-line training predictor are demonstrated through the numerical simulations of two ship maneuvers. The on-line training neural network predictor/controller is applied to the dynamic positioning (station-keeping) of a ship in a uniform current with and without external environmental disturbances. The results of the numerical simulations are very satisfactory.

Key Words: On-line Training, Neural Network Predictor/controller, Dynamic Positioning

1. Introduction

The role of dynamic positioning (DP) of offshore floating structures is becoming increasingly important when exploration and production for natural resources in oceans get into deeper water. A dynamic positioning system employs the propellers, lateral thrusters, and other control devices mounted on a structure and commanded by either human operator or automatic control system to counteract environmental forces due to wind, waves and current. The DP system maintains or assists to maintain the structure as close as possible to a desired position and heading so that the structure can operate properly. DP has many advantages over other position keeping methods (e.g. mooring lines, tension legs, etc.). These advantages (e.g. costs not increasing proportional to water depth, better accuracy, larger flexibility and wider applications, etc.) can be particularly significant for deep water operations [1].

Most automatic DP systems employ closed-loop control in which the structure's response is compared to the desired position and heading and the difference is fed back to the controller to generate the control actions. Environmental sensors and a position reference

system are needed to provide the feedback on the structure's location and heading, and environmental conditions (wind speed and direction; wave amplitude, frequency and direction; and current speed and direction; etc.).

Many traditional control algorithms require a mathematical model of the structure's motion, and a solution method to solve the equations of motion and predict the position (including orientation) of the structure given the environmental conditions. The predictions are compared continuously with the desired structure's position. The error of control action generated by a PID (proportional-integral-differential) controller is proportional to the error of the prediction of the desired value in the structure's position. The effectiveness and performance of the PID controller heavily depends on the accuracy of the mathematical model and the solution method, the computational speed of the prediction, the choice of the proportional, integral and differential constants. The computing power required for the prediction increases dramatically as the complexity and accuracy of the mathematical model increase. Besides, many PID controllers are very frequency sensitive and a filtering of the signals from the sensors is required to achieve a good control. This requires significant additional computing power. A compromise between the accuracy of the prediction and a fast response of the controller has to be made. Although significant amount of work has been devoted to increase the accuracy and speed of analytical predictions, the problem has not been solved satisfactorily.

Recently, there has been an increasing growth in interest in artificial neural networks and fuzzy logic theories over a wide spectrum of research domains. Within control engineering, neural networks and fuzzy logic controls are attractive because they hold the promise of solving problems that have so far been difficult to handle with traditional analytical methods. To acquire the knowledge about a process, a traditional analytical method uses the so-called white-box approach. If the characteristics of all the elements in the box (representing the process being considered) are known, then the relation between the output and the input of the process can be obtained. The white-box approach attempts to describe the elements in terms of mathematical formulae relating the system output to the input. Major drawbacks of the white-box approach are: 1) that a good mathematical model depends on our knowledge about each element in the box and our knowledge about the elements is incomplete most of time; and 2) that even if we can develop an accurate model, our ability to solve the mathematical problem is limited; assumptions about the physics of the process and approximations must be made to simplify the formulae. Therefore, in most cases, our knowledge about the process acquired through the white-box approach is incomplete. Neural network and fuzzy logic modeling, on the other hand, acquires the knowledge about the process using the so called black-box approach. By observing enough number of input-output samples, the neural network or fuzzy logic modeling can establish the relation between the input and output of the process using network computing or fuzzy reasoning. One of the advantages of controllers based on neural network and fuzzy logic model is that no mathematical modeling of the process is necessary, thus greatly reducing the computational time for the system behavior prediction. A considerable body of work has also shown that the neural network and fuzzy modeling can be more accurate than traditional analytical methods, especially for complicated processes.

In the recent years, attempts have been made in applying fuzzy logic controllers to surface ship path control [9], to depth control of unmanned undersea vehicles [2] and to autopilot design optimization [10]. It has been shown that fuzzy logic controllers have many advantages and can perform better than traditional controllers. A neural network controller has two main components: a process emulator (predictor) and a control algorithm. The emulator predicts the system's behavior that is used by the control algorithm to generate the control action. In a traditional controller, the emulator consists of a mathematical model and the solution method. In the neural network controller, the emulator is an artificial neural network that possesses the knowledge about the system acquired through training and can predict the behavior of the system. A set of the measured system's input-output samples are used to train the network. Once trained, the network is able to predict the system's behavior. The prediction is then used by the control algorithm to generate the control action.

The core of a fuzzy logic controller is the fuzzy associative memory rules that are derived

from expert knowledge or experience about the system. Given the control input, the controller applies appropriate rules to generate the control output. The control algorithm is simple and a fast and effective control action can be generated. Obviously, the performance of the fuzzy controller highly depends on how good the expert knowledge is. Neural network controllers have also been applied to ship maneuvering controls. [7] developed a quick adaptive method based on a neural network for control of autonomous underwater vehicles. [11] used neural network approaches to design course-keeping autopilots, track-keeping controllers and automatic berthing systems. [4,5,6] and [8] investigated a functional-link neural network for dynamic positioning of ships. Their work has shown a very promising potential for successful application of the neural networks to control of motions of offshore structures.

In this paper, we present a neural network predictor/controller for dynamic positioning of offshore structures. Our neural network predictor is developed based on the functional-link network proposed by [4,5]. However, many improvements are made to increase the accuracy and speed of the prediction of the system behavior. The major improvements include: on-line network training and cost function for optimal control, etc.

2. Neural Network Predictor

2.1 Basic Neural Network Predictor

Consider a time varying multiple-input multiple-output (MIMO) process as shown in Figure 1, where

$\mathbf{U}(t)=\{U_1(t),U_2(t),\cdots\cdots,U_N(t)\}$ -N input vector

$\mathbf{Y}(t)=\{Y_1(t),Y_2(t),\cdots\cdots,Y_M(t)\}$ -M output vector

A basic artificial neural network representation of the system is illustrated in Figure 2. The network consists of a few layers of nodes (neurons). The nodes in each layer on the left receive signals and pass the modified (or weighted) signals to the nodes in the layer on the right. The first layer (input layer) receives the input signals. The signals are then modified and passed to the next layer until the signals reach the last layer (output layer). The signals coming out from the output layer are transformed into the system's output. The layers between the input layer and the output layer are called hidden layers.

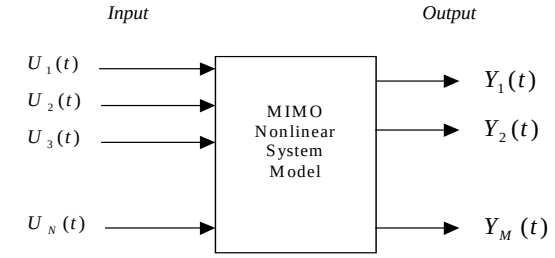

Figure 1. A MIMO system

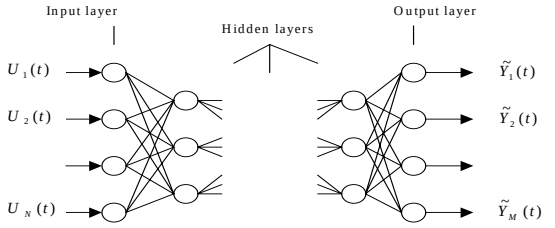

Figure 2. Structure of a basic MIMO artificial neural network

The mathematical description of the signals passing through the network can be expressed as,

$$\tilde{Y}_m(t)=g_m(O_m^I(t)) \qquad (m=1,2,\cdots\cdots,M) \tag{1}$$

where t is time, $\tilde{Y}_m(t)$ is the m^{th} component of the system output from the neural network, g_m is a squashing function. I is the total number of layers. $O_m^I(t)$ is the signal coming out from the m^{th} node in the output layer,

$$O_m^I(t)=\sum_{j=1}^{M_{I-1}} w_j^{(m,I)} O_j^{I-1}(t) \qquad (m=1,2,\cdots\cdots,M) \tag{2}$$

and the signals coming out from the m^{th} node in the i^{th} layer can be expressed,

$$O_m^i(t)=\sum_{j=1}^{M_{i-1}} w_j^{(m,i)} O_j^{i-1}(t)$$
$$(m=1,2,\cdots\cdots,M_{i-1}) \quad (i=2,3,\cdots\cdots,I-1) \tag{3}$$

with $i=1$ being the input layer and $i=I$ being the output layer. M_i is the number of nodes in the i^{th} layer, with $M_1=N$ and $M_I=M$. $O_m^1(t)=U_m(t)$ for m=1,2,….,N are the input signals. $w_j^{(m,i)}$ is a weighting constant that represents the strength of the connection between the m^{th} node in the i^{th} layer and the j^{th} node in the $(i-1)^{\text{th}}$ layer. If the weights are known, one can calculate the system output from the network using Eqs. (1-3) with the system input given.

The design of the network is to determine the weights so that the system output from the network predicts as closely the output of the real system. This is achieved by training the network with a series of measured sample input-output sets of the system. Once the network has learned the system, it can be used to predict the future behavior of the system. The network training consists of choosing the weights so that the error between the predicted system output by the network and the measured system output is minimized.

Let $(\hat{U}_n(k), n=1,2,\cdots\cdots,N)$ and $(\hat{Y}_m(k), m=1,2,\cdots\cdots,M)$ be a pair of the measured system input-output at time $t=t_k$ for $k=1,2,\cdots\cdots,K$. A cost function for training the network can be defined as functions of weighting constants,

$$f(w_j^{(m,i)}) = \sum_{k=1}^{K} \left(\sum_{r=1}^{M} \left(\tilde{Y}_r(k) - \hat{Y}_r(k) \right)^2 \right) \quad (4)$$

where $\tilde{Y}_r(k)$ is the computed network's system output by Eqs.(1-3) using the sample input $\hat{U}_n(k)$. Now the training of the network is equivalent to finding the weights to minimize the cost function.

2.2 On-line Training Functional-link Neural Network Predictor

To overcome the difficulty with the high computational cost of the basic neural network with hidden layers, [4,5] proposed use of a functional-link neural network. The structure of the network is depicted in Figure 3. For simplicity of illustration, the structure of the network for a multiple input single output (MISO) system is shown; extension to a MIMO system is straight forward.

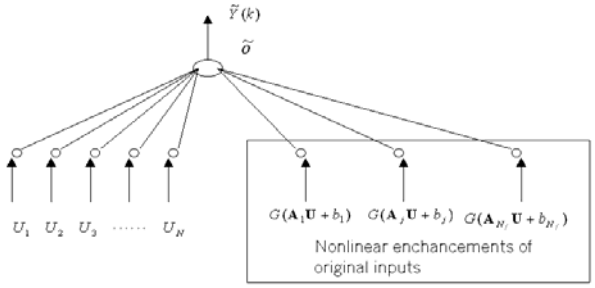

Figure 3. Structure of MISO Functional-link Neural Network

Instead of having hidden layers, Gu et al enhanced the input of the network with additional terms of nonlinear functions of the system input

$$f_j(\mathbf{U}) = G(\mathbf{A}_j \cdot \mathbf{U} + b_j) \qquad (j=1,2,\cdots\cdots,N_f) \quad (5)$$

where N_f is the number of the nonlinear enhancements. $\mathbf{A}_j$ is a vector of length N whose elements are random numbers. b_j is also a random number. g is a squashing function which usually is the hyperbolic tangent function,

$$G(u) = \tanh(u) \quad (6)$$

The network's output is expressed as,

$$\tilde{Y}(t) = G(\tilde{o}) \quad (7)$$

where

$$\tilde{o} = \sum_{n=1}^{N} w_n U_n + \sum_{j=1}^{N_f} \beta_j f_j(\mathbf{U}) \quad (8)$$

and w_n are the weights for the link of the inputs to the output and β_j are the weights for the link of the enhancements to the output.

The functional-link network can be viewed as a neural network of two layers (no hidden layer) with an expanded input vector (the original inputs plus the nonlinear enhancements). Compared to the networks with hidden layers, the number of weights to be trained is significantly reduced. [4,5], and [8] have demonstrated that the functional-link network trained with sufficient sample data from a system can serve as a universal approximator of the system, and it is fast and accurate.

However, the functional-link neural network in Figure 3 can not represent time-evolving systems very well. A time-evolving system, such as the motion of a marine vessel dynamically positioned, has memory effect. The system output depends not only on the current input but also on previous input and output. To develop a neural network for time-evolving systems, the deterministic on-line training model in a traditional parameter identification theory and the functional-link network are combined. The on-line training model assumes that the output of a time-evolving system can be expressed in the discretized form as a nonlinear function of the current input and previous input and output,

$$\mathbf{Y}(k) = \mathbf{f}(\mathbf{U}(k), \mathbf{U}(k-1), \mathbf{U}(k-2), \cdots, \mathbf{U}(k-m_u); \mathbf{Y}(k-1), \mathbf{Y}(k-2), \cdots, \mathbf{Y}(k-m_y)) \quad (9)$$

where m_u and m_y are the integers representing the time span of the memory effect for the input and output respectively. We take the system's previous input and output as additional enhancements to the functional-link

network. The structure of the new on-line training functional-link network is depicted in Figure 4. The outputs of the network at time k are

$$\tilde{Y}_m(k)=g(\tilde{o}_m(k)) \qquad (m=1,2,\cdots\cdots,M) \tag{10}$$

where

$$\begin{aligned}\tilde{o}_m(k)=&\sum_{r=0}^{m_u}\sum_{n=1}^{N} w_u(m,n,r)\,U_n(k-r)\\&+\sum_{q=1}^{m_y}\sum_{l=1}^{M} w_y(m,l,q)\,Y_l(k-q)\\&+\sum_{j=1}^{N_f}\beta_j\, g(\mathbf{A}_j\cdot\mathbf{U}(k)+b_j)\end{aligned} \tag{11}$$

and $w_u(m,n,r)$ is the weight for the link of the n^{th} component of the input at time k-r to the m^{th} component of the output at time k; $w_y(m,l,q)$ is the weight for the link of the l^{th} component of the previous output at time k-q to the m^{th} component of the output at time k; and β_j is the weights of the j^{th} nonlinear functional-link enhancement. These weights are to be trained with the sample data of the system so that the cost function is minimized,

$$f(w_u(m,n,r),w_y(m,l,q),\beta_j)=\sum_{k=1}^{K}\left(\sum_{p=1}^{M}(\tilde{Y}_p(k)-\hat{Y}_p(k))^2\right)$$

$$(\text{for } n=1,2,\cdots,N;\ m=1,2,\cdots,M;\ r=1,2,\cdots m_x;$$
$$l=1,2,\cdots,M;\ q=1,2,\cdots,m_y,\ j=1,2,\cdots,N_f) \tag{12}$$

The on-line training will be used because of its advantages discussed earlier. A set of $K+\max(m_u,m_y)$ latest data samples is used to train the network. Every time step, the oldest sample is discarded and a most recent sample is added to the data set. The network is trained every time step. Once trained, the network can be used to predict the system output for the next time step given the input at the next time step.

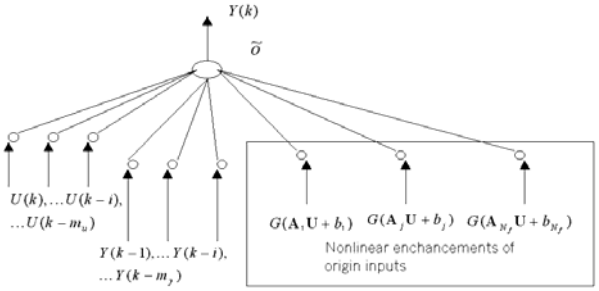

Figure 4. Structure of a SISO On-line Training Functional-link Neural Network

2.3 The System

The on-line training functional-link neural network is tested with the prediction of the motion of a maneuvering ship. Without losing generality, the results of numerical simulation of the motion of the ship, instead of the measured data of a real ship, are used for the network training and verification of the network. The purpose of the test is to verify the ability of the network to learn an unknown system, whether it is a real physical system or a numerical system. If the network can learn the numerical system, there is no reason not to believe that the network can learn the real system prediction.

Our system is based on the 3-degree-of-freedom nonlinear ship maneuvering equations given in [3],

$$\begin{aligned}(\Delta-X_{\dot u})\dot u=&X^o+X_u\delta u+\frac{1}{2}X_{uu}\delta u^2+\frac{1}{6}X_{uuu}\delta u^3+\frac{1}{2}X_{vv}v^2\\&+\frac{1}{2}X_{rr}r^2+\frac{1}{2}X_{\delta\delta}\delta_R^2+\frac{1}{2}X_{vvu}v^2\delta u+\frac{1}{2}X_{rru}r^2\delta u\\&+\frac{1}{2}X_{\delta\delta u}\delta_R^2\delta u+(X_{vr}+\Delta)vr+X_{v\delta}v\delta_R+X_{r\delta}r\delta_R\\&+X_{vru}vr\delta u+X_{v\delta u}v\delta_R\delta u+X_{r\delta u}r\delta_R\delta u\end{aligned} \tag{13}$$

$$\begin{aligned}(\Delta-Y_{\dot v})\dot v=&Y^o+Y_u^o\delta u+Y_{uu}^o\delta u^2+Y_v v+\frac{1}{6}Y_{vvv}v^2+\frac{1}{2}Y_{vrr}vr^2\\&+\frac{1}{2}Y_{v\delta\delta}v\delta_R^2+Y_{vu}v\delta u+\frac{1}{2}Y_{vuu}v\delta u^2+(Y_r-\Delta)r+\frac{1}{6}Y_{rrr}r^3\\&+\frac{1}{2}Y_{rvv}rv^2+\frac{1}{2}Y_{r\delta\delta}r\delta_R^2+Y_{ru}r\delta u+\frac{1}{2}Y_{ruu}r\delta u^2+Y_\delta\delta_R\\&+\frac{1}{6}Y_{\delta\delta\delta}\delta_R^3+\frac{1}{2}Y_{\delta vv}\delta_R v^2+\frac{1}{2}Y_{\delta rr}\delta_R r^2+Y_{\delta u}\delta_R\delta u\\&+\frac{1}{2}Y_{\delta uu}\delta_R\delta u^2+Y_{vr\delta}vr\delta_R\end{aligned} \tag{14}$$

$$\begin{aligned}-N_{\dot v}\dot v+(I_z-N_{\dot r})\dot r=&N^o+N_u^o\delta u+N_{uu}^o\delta u^2+N_v v\\&+\frac{1}{6}N_{vvv}v^3+\frac{1}{2}N_{vrr}vr^2+\frac{1}{2}N_{v\delta\delta}v\delta_R^2+N_{vu}v\delta u\\&+\frac{1}{2}N_{vuu}v\delta u^2+N_r r+\frac{1}{6}N_{rrr}r^3+\frac{1}{2}N_{rvv}rv^2\\&+\frac{1}{2}N_{r\delta\delta}r\delta_R^2+N_{ru}r\delta u+\frac{1}{2}N_{ruu}r\delta u^2+N_\delta\delta_R\\&+\frac{1}{6}N_{\delta\delta\delta}\delta_R^3+\frac{1}{2}N_{\delta vv}\delta_R v^2+\frac{1}{2}N_{\delta rr}\delta_R r^2\\&+N_{\delta u}\delta_R\delta u+\frac{1}{2}N_{\delta uu}\delta_R\delta u^2+N_{vr\delta}vr\delta_R\end{aligned} \tag{15}$$

All the quantities in the above equations are non-dimensionalized based on the ship's length, water density and the ship's initial speed moving ahead along a straight line. Two coordinate systems are used to describe the ship's motion: One is the ground-fixed system and the other ship-fixed, shown in Figure 5. The meanings of the notations are,

u ---- ship speed in x-direction of the ship-fixed system;

v ---- ship speed in y-direction of the ship-fixed system;

r ---- ship's rotation velocity (yaw rate); $r=\dot{\psi}$ where ψ is the ship's yaw angle;

$\dot{u}$ ---- acceleration of ship in x-direction of the ship-fixed system;

$\dot{v}$ ---- acceleration of ship in y-direction of the ship-fixed system;

$\dot{r}$ ---- rotational acceleration ship;

δ_R ---- rudder angle;

X^o --- increase in propeller thrust relative to the propeller thrust when the ship travels ahead steadily.

$\delta u=u-u_1$; $u_1=1$ is the non-dimensional ship's initial steady speed.

Y^o --- hydrodynamic side force on the ship when the ship travels ahead steadily.

N^o --- hydrodynamic moment on the ship about a vertical axis through the center of gravity of ship (z-axis) when the ship travels ahead steadily.

Δ --- mass of ship;

I_z --- mass moment of inertia about the z-axis.

Y^o , N^o , and the other hydrodynamic derivatives are also time-independent and can be obtained either by model tests or theoretical calculations. A dot above a variable indicates its derivative with respect to time.

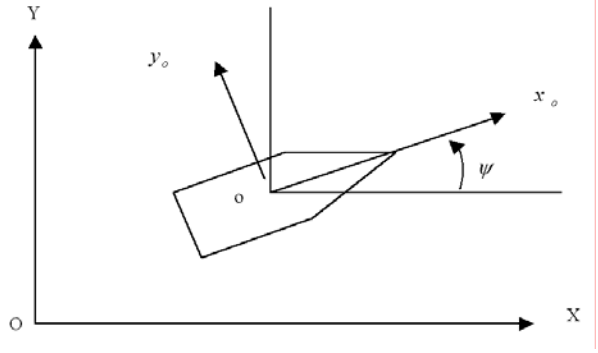

Figure 5. Coordinate systems

The ship speed in the ground-fixed system can be written in terms of u, v, and ψ,

$$\dot{x}_g = u\cos\psi - v\sin\psi \tag{16}$$

$$\dot{y}_g = u\sin\psi + v\cos\psi \tag{17}$$

where (x_g, y_g) is the position of the ship's center of gravity in the ground-fixed coordinate system; and, by definition, and we have,

$$\dot{\psi} = r \tag{18}$$

Eqs. (13-18) can be re-arranged into the following form,

$$\frac{d}{dt}\begin{Bmatrix} x_g \\ y_g \\ \psi \\ u \\ v \\ r \end{Bmatrix} = \begin{Bmatrix} h_1(x_g, y_g, \psi, u, v, r; \delta_R, X^o) \\ h_2(x_g, y_g, \psi, u, v, r; \delta_R, X^o) \\ h_3(x_g, y_g, \psi, u, v, r; \delta_R, X^o) \\ h_4(x_g, y_g, \psi, u, v, r; \delta_R, X^o) \\ h_5(x_g, y_g, \psi, u, v, r; \delta_R, X^o) \\ h_6(x_g, y_g, \psi, u, v, r; \delta_R, X^o) \end{Bmatrix} \tag{19}$$

Eq. (19) is a system of the 1st-order ODEs with respect to time. The vector $(x_g, y_g, \psi, u, v, r)$ can be regarded as the state variables because once they are determined, the system is completely known. The h functions on the right hand side of Eq. (19) are known functions of the state variables, the rudder angle δ_R, and the propeller thrust increase X^o. With (δ_R, X^o) specified and $(x_g, y_g, \psi, u, v, r)$ known at time *t*, Eq.(19) can be integrated in time to give the system output at the next time instant $t+dt$.

2.4 Prediction of Ship Motion by On-line Training Neural Network Predictor

The ship used for the numerical simulation is a Mariner class model. The information about the model can be found in [3]. In the numerical tests, the ship initially moves at a constant speed along a straight line. Then some specified rudder angle and thrust change are applied. The ship maneuver can then be simulated by solving Eq.(19) in a time-stepping

fashion. This is the process that the on-line training neural network is to learn and predict.

We first run the numerical simulation for $K+\max(m_u, m_y)$ steps and store the system input (δ_R, X^o) and output $(x_g, y_g, \psi, u, v, r)$ at every time step. So, we have $K+\max(m_u, m_y)$ "measured" samples which are used to train the neural network. Given the system input (δ_R, X^o) at the next time step, we use the trained network to predict the system's output $(x_g, y_g, \psi, u, v, r)$ at the new time step. The system output is also obtained using the numerical simulation. The network prediction is assessed by comparing it with the result of the numerical simulation. The training data is then updated by discarding the oldest set of (δ_R, X^o) and $(x_g, y_g, \psi, u, v, r)$, and adding the newest set. The above same procedure is repeated for the next time step. The procedure continues until the desired length of simulation is reached.

Two simulation cases are used for the tests: zigzag maneuver and turning path maneuver. In both cases, the network predictions agree very well with the numerical simulations. However, when used for the ship motion control, the network predictor sometimes fails to give the needed prediction. For example, in station-keeping of a ship, the objective is to keep the ship to a fixed point with a desired heading direction. Suppose that the ship has been brought to the desired position and the heading and the environmental disturbance has ceased. To maintain the desired position and the heading, no control action is needed, i.e. (δ_R, X^o) is zero. Suppose that this situation lasts for more than $K+\max(m_u, m_y)$ time steps. Then all the samples for the network training become identical. All the knowledge about the dynamic system is lost since the weights linking the input (δ_R, X^o) to the output $(x_g, y_g, \psi, u, v, r)$ are zero. When the environmental disturbance appears again, the network is not able to give the needed prediction and the control fails. To avoid this difficulty, we decompose the total system output into,

$$\begin{pmatrix} x_g \\ y_g \\ \psi \\ u \\ v \\ r \end{pmatrix}_{t+dt} = \begin{pmatrix} \partial x_g/\partial \delta_R \\ \partial y_g/\partial \delta_R \\ \partial \psi/\partial \delta_R \\ \partial u/\partial \delta_R \\ \partial v/\partial \delta_R \\ \partial r/\partial \delta_R \end{pmatrix}_t \left(\delta_R(t+dt)-\delta_R(t)\right) + \begin{pmatrix} \partial x_g/\partial X^o \\ \partial y_g/\partial X^o \\ \partial \psi/\partial X^o \\ \partial u/\partial X^o \\ \partial v/\partial X^o \\ \partial r/\partial X^o \end{pmatrix}_t \left(X^o(t+dt)-X^o(t)\right) + \begin{pmatrix} \bar{x}_g \\ \bar{y}_g \\ \bar{\psi} \\ \bar{u} \\ \bar{v} \\ \bar{r} \end{pmatrix}_{t+dt} \tag{20}$$

where the first and second terms are the linear contributions due to the changes in the rudder angle and the thrust increase, respectively. The last term contains everything left including the nonlinear contributions. The derivatives (columns in the first and second terms) are evaluated at time t and they can further be expressed as,

$$\begin{pmatrix} \partial x_g/\partial \delta_R \\ \partial y_g/\partial \delta_R \\ \partial \psi/\partial \delta_R \\ \partial u/\partial \delta_R \\ \partial v/\partial \delta_R \\ \partial r/\partial \delta_R \end{pmatrix}_t = \begin{pmatrix} \partial x_{og}/\partial \delta_R \cos\psi - \partial y_{og}/\partial \delta_R \sin\psi \\ \partial x_{og}/\partial \delta_R \sin\psi + \partial y_{og}/\partial \delta_R \cos\psi \\ \partial \psi/\partial \delta_R \\ \partial u/\partial \delta_R \\ \partial v/\partial \delta_R \\ \partial r/\partial \delta_R \end{pmatrix}_t \tag{20-1}$$

and

$$\begin{pmatrix} \partial x_g/\partial X^o \\ \partial y_g/\partial X^o \\ \partial \psi/\partial X^o \\ \partial u/\partial X^o \\ \partial v/\partial X^o \\ \partial r/\partial X^o \end{pmatrix}_t = \begin{pmatrix} \partial x_{og}/\partial X^o \cos\psi - \partial y_{og}/\partial X^o \sin\psi \\ \partial x_{og}/\partial X^o \sin\psi + \partial y_{og}/\partial X^o \cos\psi \\ \partial \psi/\partial X^o \\ \partial u/\partial X^o \\ \partial v/\partial X^o \\ \partial r/\partial X^o \end{pmatrix}_t \tag{20-2}$$

Derivatives $\left(\partial x_{og}/\partial\delta_R, \partial y_{og}/\partial\delta_R, \partial\psi/\partial\delta_R, \partial u/\partial\delta_R, \partial v/\partial\delta_R, \partial r/\partial\delta_R\right)$ are the increases in $(x_{og}, y_{og}, \psi, u, v, r)$ due to the change in the rudder. (x_{og}, y_{og}) is the movement of the center of gravity in the ship-fixed coordinate system. Similarly, derivatives $\left(\partial x_{og}/\partial X^o, \partial y_{og}/\partial X^o, \partial\psi/\partial X^o, \partial u/\partial X^o, \partial v/\partial X^o, \partial r/\partial X^o\right)$ are the increases in $(x_{og}, y_{og}, \psi, u, v, r)$ due to the change in the propeller thrust. These derivatives are not functions of time. They only depend on the ship's mass, moment of inertia and the hull geometry and can be determined by either model test or theoretical calculation. They are considered known values.

The first two terms on the right hand side of Eq.(20) are known. Only the last term needs to be learned by the neural network. The same on-line training procedure described above can be used except that the sample data must be modified by subtracting the first two terms on the right hand side of Eq. (20) from the "measured" system output $(x_g, y_g, \psi, u, v, r)$. The trained network can then predict $(\bar{x}_g, \bar{y}_g, \bar{\psi}, \bar{u}, \bar{v}, \bar{r})$, and hence $(x_g, y_g, \psi, u, v, r)$, at the next time step given the input (δ_R, X^o).

The on-line training neural network predictor using the output-decomposition treatment is now able to give a needed prediction that a control algorithm can use to generate the control action when the above-mentioned situation in the station-keeping problem occurs.

2.4.1 Zigzag Maneuver

In the zigzag maneuver, the typical procedure of conducting the test is as follows [3],

(a) Bring the ship to the steady state.

(b) Deflect the rudder at a maximum rate to a pre-selected angle and hold until a pre-selected change of heading angle is reached.

(c) At this point, deflect the rudder at the maximum rate to an opposite angle and hold until the execute change of the heading angle on the opposite side is reached. This completes the overshoot test.

(d) If a zigzag test is to be completed, again deflect the rudder at the maximum rate to the same angle in the first direction. This cycle can be repeated through the third or more executes.

For our test, the pre-selected rudder angle is 15^o and the pre-selected change of the heading angle is 10^o. The neural network parameters used are: $K = 15$, $m_u = m_y = 5$, and $N_f = 5$. The ship's initial position is at (0,0) with a zero yaw angle and moves with speed $u_1 = 1$.

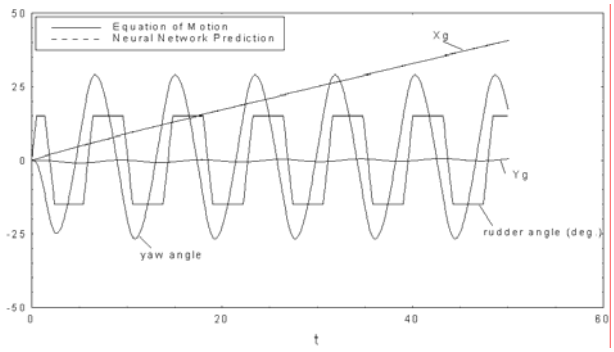

Figure 6. Rudder angle (δ_R), ship position (x_g, y_g), and yaw angle (ψ) vs. time

The rudder angle δ_R, the ship position (x_g, y_g) and yaw angle ψ (solid lines) by solving Eq. (19) and those predicted by the neural network (dashed lines) are shown in Figure 6. The neural network predictions are so close to those of Eq. (19) that one can not distinguish them graphically. Figure 7 compares the ship's trajectory by Eq. (19) and that predicted by the neural network, while Figure 8 compares the translation and rotation velocities of the ship by Eq. (19) and those by the network prediction. Again, the results from Eq. (19) and the network prediction are too close to distinguish from the figures.

Figure 9 and Figure 10 show the differences (errors) in y_g and ψ between the neural network predictions and the simulation results using Eq. (19) since these two quantities are of primary interest in the zigzag maneuver test. The average error in $|y_g|$ is 0.29% of the ship length with a maximum error of around 1%. The average

error in $|\psi|$ is 0.0038 radian with a maximum of approximately 0.045 radian. The maximum error in ψ occurs every time when $|\psi|$ reaches the maximum, and stays for only a very short period of time.

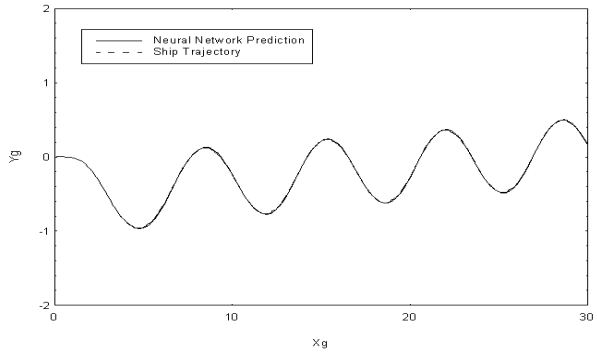

Figure 7. Trajectory of the Ship's center of gravity

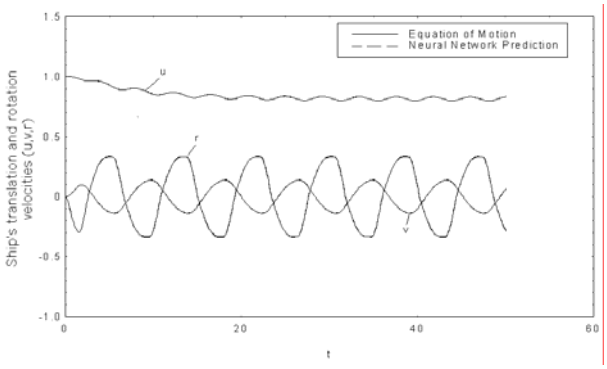

Figure 8. Translation and rotation velocity of the ship vs. time

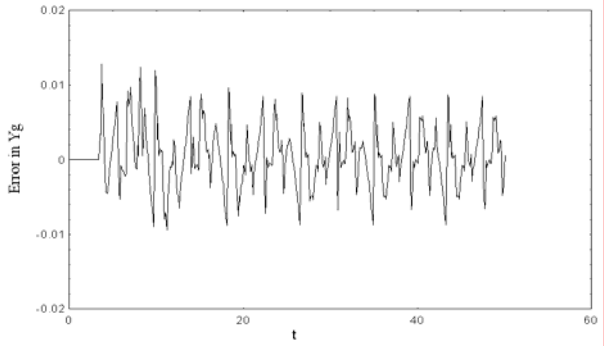

Figure 9. Error in y_g vs. time t

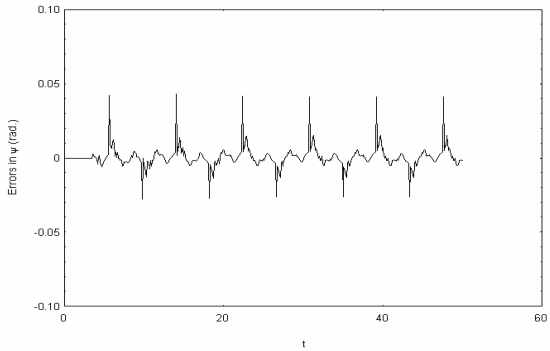

Figure 10. Error in ψ vs. time t

2.4.2 Turning Path Maneuver

Turning path maneuver is another important test of the ship's maneuverability, and is also used to assess the effectiveness of the neural network. The same network parameters and ship's initial condition as those in the zigzag maneuver are used. The rudder angle is deflected at the maximum rate to a pre-selected maximum angle of 10^o and then held unchanged during the whole simulation period.

Figure 11 compares the trajectory of the ship's center of gravity by Eq. (19) and the network prediction. The two are too close to see the difference. Figure 12 shows the translation and rotation velocities of the ship of the results of the simulation and the network prediction. As expected, they are almost identical. Figure 13 shows the errors in x_g, y_g and ψ. The average error in $|x_g|$ is 0.16% with a maximum error of 1.5%. The average error in $|y_g|$ is 0.15% with a maximum error of 3%. The average error in $|\psi|$ is 0.0016 radian with a maximum error of 0.008 radian. The maximum error occur during the early stage of the maneuver when the rudder is deflected from zero to 10^o rapidly.

Figure 11. Trajectory of the ship's center of gravity (turning path maneuver)

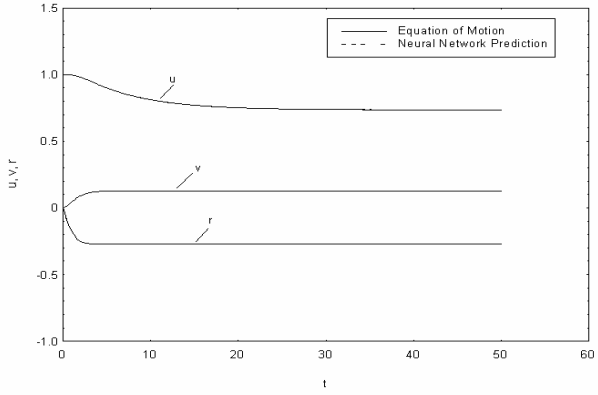

Figure 12. Ship's translation and rotation velocities (turning path maneuver)

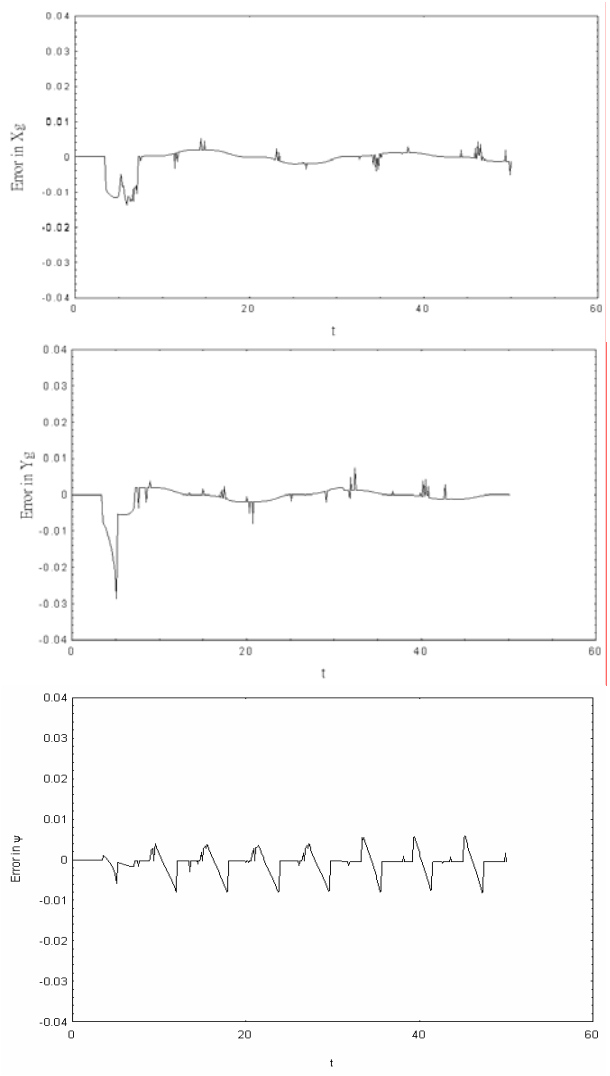

Figure 13. Errors in x_g, y_g and ψ vs. time t

One thing worth pointing out is that the errors in the neural network predictions are bounded with respect to time because of the on-line training. The numerical tests have shown that the neural network is fast and the neural network predictions are satisfactory.

3. Neural Network Controller

We combine the on-line training neural network predictor with an optimal controller to result in a neural network controller for dynamic positioning of a ship. The control objective is to bring the ship to a desired fixed point (x_d, y_d) in the ground-fixed coordinate system and keep the ship in that position with a desired heading ψ_d. We define a cost function,

$$\Phi(\delta_R(t+dt), X^o(t+dt)) = (\tilde{x}_g(t+dt)-x_d)^2 + (\tilde{y}_g(t+dt)-y_d)^2 + (\tilde{\psi}(t+dt) - \breve{\psi}_d(t))^2 + C_1\left[(\tilde{u}(t+dt)-u_1)^2 + \tilde{v}^2(t+dt)\right] + C_2\tilde{r}^2(t+dt) + C_3(\delta_R(t+dt)-\delta_R(t))^2 \quad (21)$$

where $(\delta_R(t+dt), X^o(t+dt))$ is the rudder angle and the propeller thrust increase to be determined at time $t+dt$. The optimal control algorithm searches for an optimal control action $(\delta_R(t+dt), X^o(t+dt))$ so that the cost function is minimized. In Eq. (21), variables with a "~" on the top are the predicted system output at $t+dt$. $\breve{\psi}_d(t)$ is a desired intermediate heading angle defined as,

$$\breve{\psi}_d(t) = \psi_d - arctg\left(\frac{y_g(t)}{X_{ahead}}\right) \quad (22)$$

The first three terms on the right hand side of Eq.(21) aim at bringing the ship to the desired position and heading, while the remaining terms are introduced to reduce the excessive overshooting and high frequency oscillatory action of the rudder angle. The desired intermediate heading is also used for this purpose. Constants C_1, C_2 and C_3 are parameters to determine the relative importance between the first three terms and the remaining terms in Eq. (21). X_{ahead} is some length usually about 4-5 ship lengths.

For practical ship control, there are constraints on how fast the rudder can move and how fast the propeller thrust can change. There are also constraints on the maximum rudder angle and the maximum propeller thrust. Therefore, the control algorithm searches the optimal $(\delta_R(t+dt), X^o(t+dt))$ by minimizing the cost function with the constraints on δ_R and X^o. For the numerical simulations presented in this paper, the constraints are set as: the maximum δ_R is 30^o, the maximum change in δ_R in one time step is 2^o; The maximum X^o is 0.05, and the maximum change in X^o in one time step is 0.005.

The performance of the controller not only depends on the accuracy of the prediction but also on the control algorithm and the associated

cost function. To verify the performance of the proposed control algorithm and the cost function, we first use the "exact" prediction from the equation-of-motion simulation to calculate the cost function. Our tests have shown that the performance of the controller is very satisfactory. We then use the neural network prediction and examine the performance of the on-line training neural network controller.

Eq. (19) is used to calculate the ship's motion in the simulation of the dynamic positioning of the ship with the neural network controller. At time *t*, the network is trained with the sample data calculated using Eq. (19) at previous time steps. The system's status $(x_g, y_g, \psi, u, v, r)_{t+dt}$ predicted by the trained network is used to evaluate the cost function in searching for the optimal $(\delta_R(t+dt), X^o(t+dt))$. This optimal control action is then used in Eq. (19) to calculate the status of the ship at $t+dt$. The rudder angle and the thrust are assumed to vary linearly from ($\delta_R(t), X^o(t)$) to ($\delta_R(t+dt), X^o(t+dt)$). The results of the numerical simulations of three control problems are presented.

3.1 Case 1 (without environment disturbance)

The first case is a ship initially located at (0,4) against a current in the –x direction in the ground coordinate frame. No environmental disturbances are present. The task is to bring the ship to point (5,0) and keep it in the position with a zero heading angle. Notice that Eq.(19) can be used to simulate the ship motion in a current with a simple coordinate transformation.

Figure 14 shows the path and the heading of the ship. Figure 15 and Figure 16 show the rudder angle and the thrust increase as functions of time. The coordinates of the ship's center of gravity and its heading are given in Figure 17. These figures clearly demonstrate that the neural network controller has successfully accomplished the control task.

3.2 Case 2 (with environment disturbance)

The second case is the same as the first case except that there are some environmental disturbances present. The resultant effects of the disturbances can be represented by a force and an moment acting on the ship which are added to Eq. (19) in the numerical simulation. For the results presented, the disturbing force and moment are

$$\begin{cases} F_x = 0.0001\sin(\omega_1 t) + 0.0002\sin(\omega_2 t + \pi/6) \\ F_y = 0.0002\sin(\omega_1 t) + 0.0004\sin(\omega_2 t + \pi/6) \\ M_g = 0.00005\sin(\omega_1 t) + 0.0001\sin(\omega_2 t + \pi/6) \end{cases} \tag{23}$$

with $\omega_1 = 1$ and $\omega_2 = 2$.

Figure 18 shows the path and the heading of the ship. Figure 19 and Figure 20 show the optimal δ_R and X^o as functions of time. The coordinates of the ship's center of gravity and its heading are given in Figure 21. As expected, they are different from those in the first case during the transient period because of the external disturbances. But the neural network controller is able to bring the ship to the desired position within about a same amount of time and keep the ship in the position with the desired heading.

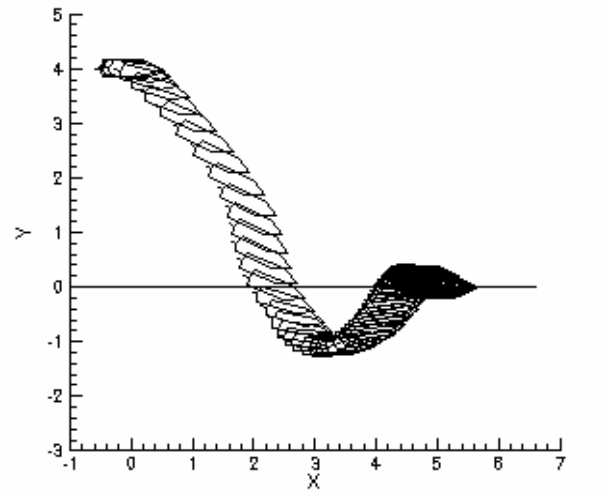

Figure 14. Path and heading of ship (no disturbance)

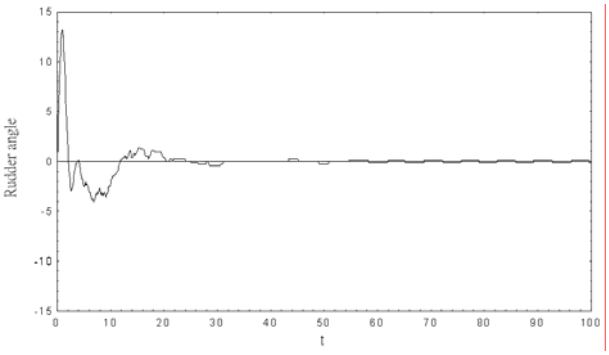

Figure 15. Optimal rudder angle vs. time (no disturbance)

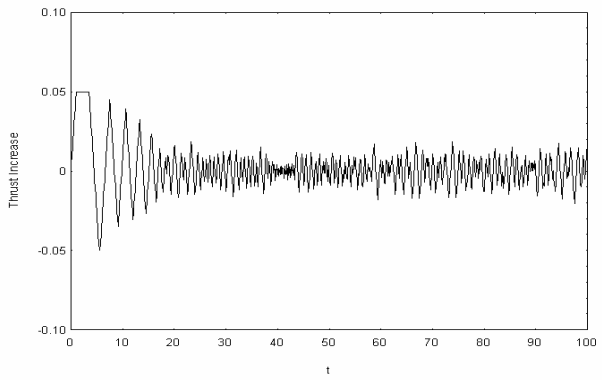

Figure 16. Optimal thrust increase vs. time (no disturbance)

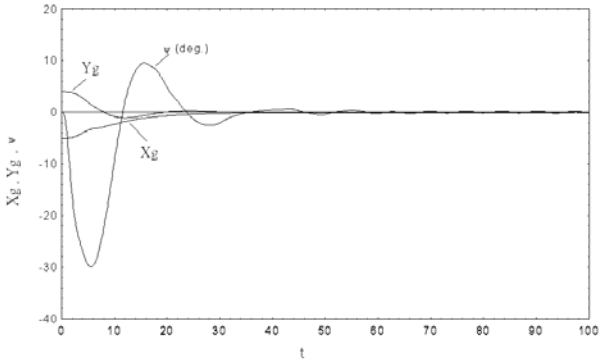

Figure 17. Coordinates of ship's center and heading angle vs. time (no disturbance)

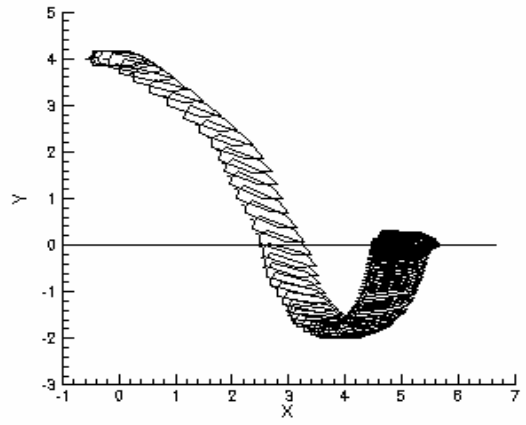

Figure 18. Path and heading of ship (with disturbance)

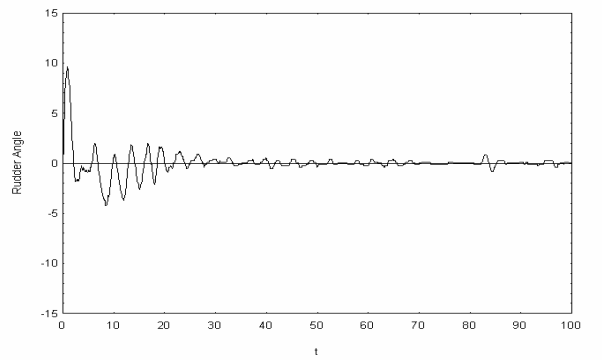

Figure 19. Optimal rudder angle vs. time (with disturbance)

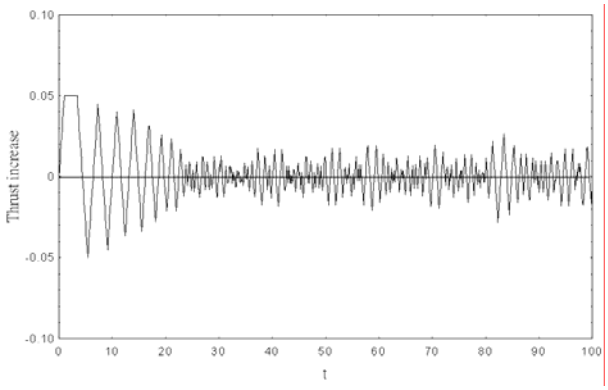

Figure 20. Optimal thrust increase vs. time (with disturbance)

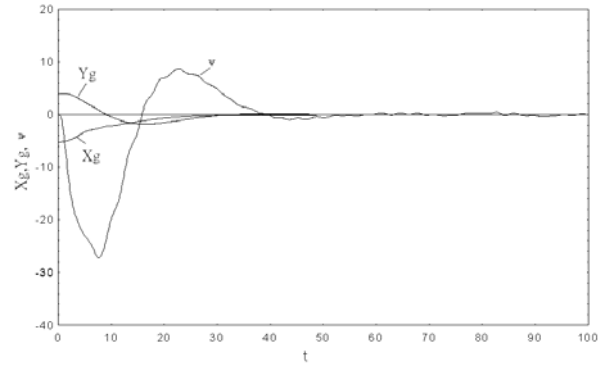

Figure 21. Coordinates of the ship's center of gravity and heading angle vs. time t (with disturbance)

4. Conclusion and Discussion

In this paper, we describe an on-line training neural network predictor for prediction of the output of MIMO systems. The advantages of the structure of the on-line training neural network and the on-line training of the network are discussed. A procedure of training the network on-line and utilizing the trained network with an optimal control algorithm for dynamic positioning (station keeping) of a ship is presented.

The effectiveness and accuracy of the network predictor are verified with the numerical ship maneuvering simulations: zigzag maneuver and turning path maneuver. The simulations have shown that the on-line training neural network is effective and fast. The network prediction is very accurate.

The feasibility of the neural network predictor/controller for dynamic positioning of marine vessels is demonstrated with the numerical simulations of station keeping of a ship with and without environmental disturbances. The neural network controller successfully accomplished the station-keeping task. The predictor/controller was able to adapt itself through the on-line training and cope with the new situations (such as environmental disturbances) without the need for

additional means to measure and analyze the disturbances.

The on-line training neural network predictor was designed for a general MIMO system so it can have a wide range of applications. In principle, this neural network predictor can be applied to any MIMO system as long as the system's input and output are measurable. In this paper, the predictor has been validated with a 2-input, 6-output system.

It is believed that, once validated, the on-line training neural network predictor/controller can be a very powerful, robust and reliable tool for floating structure operators.

Acknowledgment

The authors wish to thank the referees for their valuable suggestions and comments that improved the presentation. This work was supported in part by the National Science Council of ROC under grant NSC 89-2611-E-032-002.

Nomenclature

$\mathbf{A}_j$ a vector of length N whose elements are random numbers.

b_j random number.

g_m squashing function.

I_z mass moment of inertia about the z-axis.

I the total number of layers.

$O_m^I(t)$ the signal coming out from the m^{th} node in the output layer.

M_i the number of nodes in the i^{th} layer.

N_f the number of the nonlinear enhancements.

N^o hydrodynamic moment on the ship about a vertical axis through the center of gravity of ship (z-axis) when the ship travels ahead steadily.

r ship's rotation velocity (yaw rate).

$\dot{r}$ rotational acceleration ship.

t time.

u ship speed in x-direction of the ship-fixed system.

$\dot{u}$ acceleration of ship in x-direction of the ship-fixed system.

v ship speed in y-direction of the ship-fixed system.

$\dot{v}$ acceleration of ship in y-direction of the ship-fixed system.

$w_j^{(m,i)}$ weighting constant.

w_n the weights for the link of the inputs to the output.

X^o increase in propeller thrust relative to the propeller thrust when the ship travels ahead steadily.

(x_{og}) is the movement of the center of gravity in the ship-fixed coordinate system.

$\tilde{Y}_m(t)$ the m^{th} component of the system output from the neural network.

(y_{og}) is the movement of the center of gravity in the ship-fixed coordinate system.

Y^o hydrodynamic side force on the ship when the ship travels ahead steadily.

δ_R rudder angle.

δu the non-dimensional ship's initial steady speed.

Δ mass of ship.

β_j the weights for the link of the enhancements to the output.

ψ the ship's yaw angle.

References

[1] CMPT, "*Floating Structures: a Guide for Design and Analysis*" (1998).

[2] DeBitetto, P. A., "Fuzzy logic for depth control of unmanned undersea vehicles," *Proc. of Symposium on Autonomous Underwater Vehicle Technology*, Cambridge, MA, U. S. A. (1994).

[3] Edward V. L., "Principles of naval architecture (PNA)," *The Society of Naval Architects and Marine Engineers*, Vol. III (1989).

[4] Gu, M. X., Pao, Y. H. and Yip, P. P. C, "Neural-net computing for dynamic positioning of vessels at sea," *Marine Jubilee Meeting*, Wageningen, The Netherlands (1992).

[5] Gu, M. X., Pao, Y. H. and Yip, P. P. C, "Neural-net computing for real-time control of a ship's dynamic positioning at sea," *Computer Engineering Practice*, pp. 305-314 (1993).

[6] Gu, M. X. and Li, D., "Dynamic positioning of ships using a 1-step ahead neural network controller," *Int. Conf. on Hydrodynamics*, Wuxi, China (1994).

[7] Ishii, K., Fujii, T. and Ura, T., "A quick adaptive method in a neural network based control system for AUVs," *Proc. of*

Symposium on Autonomous Underwater Vehicle Technology, Cambridge, MA, USA (1994).

[8] Li, D. and Gu, M.X., "Dynamic positioning of ships using a planned neural network controller," *J. of Ships Research*, Vol. 40, No. 2 (1996).

[9] Parsons, M. G, Chubb, A. C., Cao, Y. and Stefanopoulou, A. G., "An initial assessment of fuzzy logic vessel path control," *Proc. of Sym. on Autonomous Underwater Vehicle Technology*, Cambridge, MA, USA (1994).

[10] Robert, G. N., "Approaches to fuzzy autopilot design optimization," *Proc. of Manoeuvring and Control of Marine Craft* (1997).

[11] Zhang, Y., Hearn, G. E. and Sen, P., "Neural network approaches to a class of ship control (Part I: Theoretical Design)," *11th Ship Control Systems Symposium*, Vol. 1 (1997).

[12] Zhang, Y., Hearn, G. E. and Sen, P., "Neural network approaches to a class of ship control (Part II: Simulation Studies)," *11th Ship Control Systems Symposium*, Vol. 1 (1997).

Manuscript Received: Jun. 22, 2001
and Accepted: Jul.. 24, 2001